

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **11259553 A**

(43) Date of publication of application: **24 . 09 . 99**

(51) Int. Cl.

G06F 17/50
G06F 9/06

(21) Application number: **10062555**

(22) Date of filing: **13 . 03 . 98**

(71) Applicant: **OMRON CORP INABATA & CO LTD**

(72) Inventor: **KATAOKA SHOICHI**
FURUWATARI TOSHIKI

(54) **DESIGN SUPPORTING METHOD FOR SYSTEM WHERE HARDWARE AND SOFTWARE COEXIST**

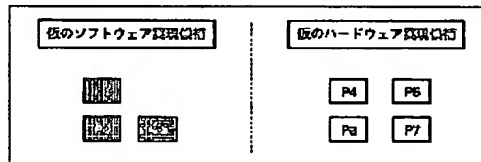
is evaluated further and thus, the division point is automatically corrected to the optimum one.

(57) Abstract:

COPYRIGHT: (C)1999,JPO

PROBLEM TO BE SOLVED: To automatically decide temporary software part and hardware part from a system specification description and then to automatically optimize the division point of both by evaluation through simulation.

SOLUTION: The system specification description for describing the specifications of a system by the set of the processes of an execution unit is prepared, and based on the system specification description, all the processes are initially divided into temporary hardware realization candidates P4-P7 and temporary software realization candidates P1-P3. At the time, information for indicating priority is attached to the one accompanying the priority in the execution of a processing, and at the time of initial division, the ones to which the information for indicating the priority is attached are turned to the software realization candidates and the others are turned to the hardware realization candidates. Then, the candidates are simulated, execution conditions are stored as profiling information, a change candidate from the respective candidates is selected based on it, the improvement rate of a chip area and an effective speed



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-259553

(43) 公開日 平成11年(1999) 9月24日

(51) Int.Cl.⁶

G 0 6 F 17/50
9/06

識別記号

5 3 0

F I

G 0 6 F 15/60
9/06

6 5 4 M
5 3 0 U

審査請求 未請求 請求項の数 7 O L (全 11 頁)

(21) 出願番号 特願平10-62555

(22) 出願日 平成10年(1998) 3月13日

(71) 出願人 000002945

オムロン株式会社

京都府京都市右京区花園土堂町10番地

(71) 出願人 595144466

稲畑産業株式会社

大阪府大阪市中央区南船場1丁目15番14号

(72) 発明者 片岡 庄一

京都府京都市下京区木津屋橋通西洞院東入
る東塩小路町606番地 オムロンソフトウ
ェア株式会社内

(72) 発明者 古渡 俊明

東京都台東区上野3丁目3番8号 株式会
社アイ・ケイ・テクノロジー内

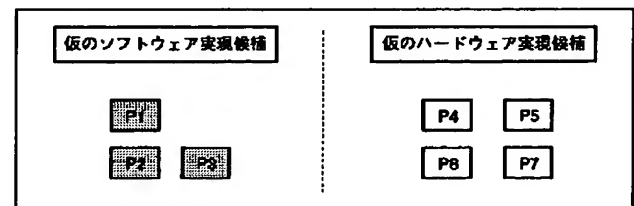
(74) 代理人 弁理士 小森 久夫

(54) 【発明の名称】 ハードウェアとソフトウェアの混在するシステムの設計支援方法

(57) 【要約】

【課題】 システム仕様記述から自動的に仮のソフトウェア部分をハードウェア部分を決め、その後シミュレートによる評価により、双方の分割点を自動的に最適化していく。

【解決手段】 システムの仕様を実行単位のプロセスの集まりで記述したシステム仕様記述を作成し、このシステム仕様記述に基づいて、全プロセスを仮のハードウェア実現候補 P 4 ~ P 7 と、仮のソフトウェア実現候補 P 1 ~ P 3 とに初期分割する。このとき、処理の実行に優先度の伴うものについては優先度を示す情報が付されており、初期分割に際しては、この優先度を示す情報の付されたものをソフトウェア実現候補とし、それ以外をハードウェア実現候補とする。次に、これらの候補をシミュレーションして実行状況をプロファイリング情報として記憶し、これに基づいてそれぞれの候補からの変更候補を選択して、さらに、チップ面積や実効速度の改善率を評価することにより、分割点を自動的に最適なものに修正する。



プロセスの初期分割

1

【特許請求の範囲】

【請求項 1】 システムの仕様を実行単位のプロセスの集まりで記述したシステム仕様記述を作成し、このシステム仕様記述に基づいて、全プロセスを仮のハードウェア実現候補と仮のソフトウェア実現候補に初期分割し、次いで、これらの候補をシミュレーションして実行状況をプロファイリング情報として収集し、このプロファイリング情報に基づいて、仮のハードウェア実現候補からソフトウェア実装への変更候補、または、仮のソフトウェア実現候補からハードウェア実装への変更候補を選択し、これらの変更候補によるチップ面積と実行速度の改善率を評価して、上記各候補の分割点を修正することを特徴とする、ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【請求項 2】 プロセスは、実行のための優先度の情報を含み、初期分割は、この優先度の含むプロセスをソフトウェア実現候補、含まないプロセスをハードウェア実現候補として自動的に行うようにした、請求項 1 記載の、ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【請求項 3】 前記変更候補の最小単位は、プロセス内の実行ブロックである、請求項 1 または 2 記載の、ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【請求項 4】 プロファイリング情報は、ソフトウェア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以上の変数が属する実行ブロックをハードウェア実装への変更候補とする、請求項 1～3 のいずれかに記載の、ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【請求項 5】 プロファイリング情報は、ハードウェア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以下の変数が属する実行ブロックをソフトウェア実装への変更候補とする、請求項 1～3 のいずれかに記載の、ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【請求項 6】 各候補の分割点を修正するとき、速度改善値のより高いハードウェア実装変更候補と、その候補と実装面積の均衡をとり得るソフトウェア実装変更候補の組とを選び、これらの候補をそれぞれハードウェア実装分、ソフトウェア実装分に変更することで分割点修正を行うようにした、請求項 1～5 のいずれかに記載の、ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【請求項 7】 各候補の分割点を修正するとき、面積改善値のより高いソフトウェア実装変更候補と、その候補と実行時間の均衡をとり得るハードウェア実装変更候補の組とを選び、これらの候補をそれぞれソフトウェア実装分、ハードウェア実装分に変更することで分割点修正を行うようにした、請求項 1～5 のいずれかに記載の、

2

ハードウェアとソフトウェアの混在するシステムの設計支援方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、ハードウェアとソフトウェアの混在するシステムの設計支援方法に関し、特に、システム仕様からハードウェアとソフトウェアの分割を自動化、且つ最適化出来る支援方法に関する。

【0002】

【従来の技術】 今日、ASIC の高集積化は数千万トランジスタの 1 チップ化を可能にしており、CPU、ファームウェア、ドライバなどに加えて、通信回路、入出力バス・インターフェイス回路などの周辺回路なども全て一つの ASIC 上に納まることを可能にしている。

【0003】 このような ASIC は、いわゆるシステム ASIC と称される。

【0004】 ところでシステム ASIC には、プロセッサとペリフェラルとメモリが実装され、プロセッサとメモリを利用して従来のソフトウェアが実装される形になる。ここで一つの機能をハードウェアで実現するかソフトウェアで実現するかにおいて、速度とチップ面積のトレードオフが内在している。つまり、ハードウェアはソフトウェアに対して相対的に速度を上げるがチップ面積も押し上げる。また、ソフトウェアはその反対の特性を持つ。

【0005】 設計に際して、重要なことは、そのトレードオフの関係を考慮し、与えられたチップ面積上で最大の速度を上げ得る双方の混在化、すなわち、ハードウェアとソフトウェアの分割点の最適化を図ることである。

【0006】 従来、この要求に対しては、熟練した人間の経験及び勘を頼りに、ある機能を実現するために、まず、ハードウェア部分とソフトウェア部分を決め、それぞれが独立して設計された後に協調動作シミュレーションなどを行って、不都合な部分を抽出し、さらに、再設計、シミュレーションを繰り返していく手法がとられている。

【0007】 また、特開平 9-160949 号では、まず、ハードウェア部分とソフトウェア部分を決めた後に、ソフトウェア部分のハードウェア化可能部分を探索し、可能な部分があれば、ソフトウェアを行うという手法が提案されている。

【0008】

【発明が解決しようとする課題】 しかしながら、上記いずれの手法も、最初に、ハードウェア部分とソフトウェア部分を人間が決められているため最終結果は必ずしも、速度とチップ面積の観点で最適なものとはならなかったり、また、ハードウェア部分とソフトウェア部分そのものを分割すること自体、極めて困難であった。また、通常、ハードウェア部分は HDL 言語（ハードウェア回路用言語）で書かれてこれをロジックで表現する一方、ソ

フトウェア部分はC言語などで書かれるため、従来の手法では両者は別扱いする必要があり、これらを協調的に同時並行開発することが基本的に無理であった。すなわち、双方の部分の交換を同時並行的に行ったり、その交換直後に結果をシミュレートする等、協調と同時並行開発を行うことは出来なかった。本発明の目的は、システム仕様記述から自動的に仮のソフトウェア部分をハードウェア部分を決め、その後シミュレートによる評価により、双方の分割点を自動的に最適化していくことの出来る、ハードウェアとソフトウェアの混在するシステムの設計支援方法を提供することにある。

【0009】

【課題を解決するための手段】本出願の請求項1に係る発明は、システムの仕様を実行単位のプロセスの集まりで記述したシステム仕様記述を作成し、このシステム仕様記述に基づいて、全プロセスを仮のハードウェア実現候補と仮のソフトウェア実現候補に初期分割し、次いで、これらの候補をシミュレーションして実行状況をプロファイリング情報として収集し、このプロファイリング情報に基づいて、仮のハードウェア実現候補からソフトウェア実装への変更候補、または、仮のソフトウェア実現候補からハードウェア実装への変更候補を選択し、これらの変更候補によるチップ面積と実行速度の改善率を評価して、上記各候補の分割点を修正することを特徴とする。

【0010】本発明では、システム仕様を、実行単位のプロセスの集合として記述することを出発点とする。この実行単位のプロセスとは、CPUで実行される並列処理単位で、且つハードウェアでもソフトウェアでも実現可能なものを言う。つまり、システム仕様記述上は、各プロセスは見かけ上、ハードウェアでもソフトウェアでも実現可能なようになっている。

【0011】次に、システム仕様記述から、ハードウェア部分とソフトウェア部分の実現候補を自動的に決めて分割する。プロセスには種々の理由からハードウェアで実現する方が望ましいものとソフトウェアで実現する方が望ましいものが存在するから、これを自動判断し、且つそれに基づきハードウェア部分とソフトウェア部分とを仮に決め、残りの部分については、ランダムに決めることが出来る。

【0012】次に、これらの仮の候補に対してシミュレーションを行い、実行状況をプロファイリング情報として収集する。シミュレーションに際しては、ハードウェア部分及びソフトウェア部分の各プロセスの実行時間、より詳細にはプロセスに含まれる実行ブロックの実行時間をクロックで計測することが出来る。これにより、実行状況としてハードウェア部分及びソフトウェア部分の各プロセスの実行速度を計測し、これをプロファイリング情報として収集することが出来る。

【0013】上記プロファイリング情報によれば、各プ

ロセスの実行速度がわかるから、実効速度の遅いソフトウェア部分のプロセスに対してはハードウェアに変更することが考えられる。この場合、ハードウェアへの変更により速度は上がるがチップ面積が増加することが実際に変更するか否かの評価対象となる。また、実行速度の遅いハードウェア部分のプロセスに対してはソフトウェアに変更することが考えられる。この場合、ソフトウェアへの変更によりチップ面積は減少するが速度が低下することが実際に変更するか否かの評価対象となる。

【0014】このようにして、変更候補によるチップ面積と実行速度の改善率を評価して、各候補の分割点を修正する。

【0015】なお、分割点の修正後、再びその分割によるハードウェア部分とソフトウェア部分でシミュレーションを行い、上記と同様な評価を繰り返すことも出来る。

【0016】本出願の請求項2に係る発明は、プロセスは、実行のための優先度の情報を含み、初期分割は、この優先度の含むプロセスをソフトウェア実現候補、含まないプロセスをハードウェア実現候補として自動的に行うようにしたものである。

【0017】プロセスは相互に通信を行いながら並行動作することが基本であるため（リアルタイム処理）、ソフトウェア部分のプロセスについては、CPUの実行権を得るための指標が予めプロセス中に含まれているのが望ましい。この請求項2に係る発明では、その指標として処理の優先度の情報を与えるものである。したがって、初期分割を行うときに、優先度の情報があるプロセスをソフトウェア実現候補として、それ以外をハードウェア実現候補として分割する。

【0018】本出願の請求項3に係る発明は、前記変更候補の最小単位は、プロセス内の実行ブロックであることを特徴とする。

【0019】1つのプロセス内には1つ以上の実行ブロックが含まれているが、その中には処理速度の高速性が要求されるものと要求されないものが混在することがある。したがって、プロセス単位を入れ替えとするよりも実行ブロック単位を入れ替え対象とする方が最適化を図る上で好ましいと言える。この場合、実行ブロック単位が最小単位であり、これ以上のかたまりを単位として扱うことも可能である。本出願の請求項4に係る発明は、プロファイリング情報は、ソフトウェア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以上の変数を含むプロセスをハードウェア実装への変更候補とするものである。また、請求項5に係る発明は、プロファイリング情報は、ハードウェア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以下の変数が属する実行ブロックをソフトウェア実装への変更候補とするものである。

【0020】システム仕様を表すプロセスはそれ自体シミュレータで実行可能な形式でないから、これを翻訳してシミュレーションコード（プログラム）を起こしてシミュレータにかけるが、このとき、シミュレータにおいて、各プロセスに含まれる実行ブロックに属する変数へのアクセス回数をプロファイリング情報として集める。そして、ソフトウェア実現候補のうちアクセス回数の多いブロックはハードウェア実装への変更候補とし、ハードウェア実現候補のうちアクセス回数の少ないブロックはソフトウェア実装への変更候補とする。すなわち、ソフトウェア部分で頻繁にアクセスするものはシステムのスループットを悪くすると推定でき、一方、ハードウェア部分であまりアクセスしないものはチップ面積の無駄遣いと推定されるから、このような部分については変更の候補とする。

【0021】本出願の請求項6に係る発明は、各候補の分割点を修正するとき、速度改善値のより高いハードウェア実装変更候補と、その候補と実装面積の均衡をとり得るソフトウェア実装変更候補の組とを選び、これらの候補をそれぞれハードウェア実装分、ソフトウェア実装分に変更することで分割点修正を行うようにしたものである。このように実装面積の均衡がとれるように、速度改善値のより高いハードウェア実装変更候補とソフトウェア実装変更候補を選択することで、面積を一定に保ちながら実行速度の点で有利な分割点を自動的に得ることが出来る。

【0022】本出願の請求項7に係る発明は、各候補の分割点を修正するとき、面積改善値のより高いソフトウェア実装変更候補と、その候補と実行時間の均衡をとり得るハードウェア実装変更候補の組とを選び、これらの候補をそれぞれソフトウェア実装分、ハードウェア実装分に変更することで分割点修正を行うようにしたものである。

【0023】このように実行時間の均衡がとれるように、面積改善値のより高いソフトウェア実装変更候補とハードウェア実装変更候補を選択することで、実行時間を一定に保ちながら実装面積の点で有利な分割点を自動的に得ることが出来る。

【0024】

【発明の実施の形態】図1は、本発明の実施形態である、ハードウェアとソフトウェアの混在するシステムの設計支援装置の構成図である。

【0025】この支援装置は、システム仕様記述1で書かれたプロセスに基づいて協調合成システム2において、ハードウェア部分とソフトウェア部分とに分割し、ハードウェア部分については動作合成システム3、論理合成システム4でHDL言語に変換しつつハードウェアロジック回路を自動作成する。また、ソフトウェア合成システム5は、ソフトウェア部分からC言語などによるプログラムコードを自動作成する。スタティックタイミ

ング検証システム6は、このプログラムコードやハードウェアロジック回路の静的動作（スタティック動作）の検証を行い。合成結果表示システムは、上記一連の手順に伴う結果を適宜表示する。

【0026】システム仕様記述のプロセスの1例を図2に示す。

【0027】ここでは、プロセス名を「sample」としている。

【0028】第2行は、整数（int）入力データ端子として、iDataを定義する。第3行は、整数出力端子として、oDataを定義し且つ初期値が0であることを示す。第4行、第5行は、制御入力端子Strtと制御出力端子ackStrtを定義し、制御出力端子ackStrtの初期値がFALSEであることを示す。第6行以下はプロセス実行部分である。要約すれば、制御入力端子StrtがTRUEのときに、制御出力端子ackStrtをTRUEにセットし、変数counterのインクリメント値が入力データ端子iDataの値よりも大きければ変数counterをリセットし、小さくなければ出力データ端子oDataの値を、iData値にcounter値を加えた値とする。

【0029】このようなプロセスは、基本的にソフトウェアでもハードウェアでも実現が可能である。

【0030】図1の協調合成システム2は、上記プロセスを読み込んで、全プロセスをハードウェア部分（ハードウェア実現候補）とソフトウェア部分（ソフトウェア実現候補）とに初期分割し、これをシミュレーションして相互の各プロセス部分のソフトウェア化またはハードウェア化が適正か否かを評価し、評価結果にしたがって、1部の入れ替えを行い、さらに、必要に応じてその入れ替えた結果で再度シミュレーションを行う動作を繰り返す。

【0031】システム仕様記述1に書かれる各プロセスは、本実施形態では初期分割しやすいように、ソフトウェア部分の候補となるプロセスに処理の優先度を表す情報がつけ加えられる。図3にその状態を示す。なお、ソフトウェアのプロセス群は1つのCPUで処理が実行される限り、各プロセス間で通信をするときに処理の優先度が必要となることがある。優先度を表す情報はこのためのものである。ハードウェアのプロセス群はCPUにより処理されるものではないから、通常は処理に優先度を必要としない。もちろん、この情報が付加されていてもこれをハードウェア部分で構成することは可能である。

【0032】図3において、PROCA0、PROCB1は、前者のプロセス優先度が「0」、後者のそれが「1」であることを示している。優先順位は前者の方が一つ高い。

【0033】協調合成システム2は、上記の優先度を表す情報が付加されているプロセスを仮のソフトウェア実現候補とする。また、その他のプロセスを仮のハードウェア実現候補とする。初期分割はこのようにして行われ

る。図4に初期分割した状態を示している。P1～P3は優先度を表す情報を持つプロセスであるため、ソフトウェア実現候補とされる。P4～P7は優先度を表す情報を持たないプロセスであるため、ハードウェア実現候補とされる。

【0034】なお、この段階で、図4のように分類された仮のソフトウェア実現候補と仮のハードウェア実現候補に対して、面積（ソフトウェア部分についてはプログラムステップ数）と、実行時間が初期見積もりデータとして保存される。これらの値は、図1のソフトウェア合成システム5と動作合成システム3により求められる。すなわち、プログラムステップ数及び面積は、ソフトウェア部分のプログラムステップ数及びハードウェア部分のHDL言語から分析した回路により、また、実行時間は、予め設定されているクロック時間と上記プログラムステップ数及び回路の遅延時間とにより求められる。

【0035】次に、上記の仮のハードウェア実現候補と仮のソフトウェア実現候補をシミュレータにかけて、実行状況のプロファイリング情報を収集する。

【0036】図5は、プロセスの一般構造を示している。システム仕様記述の中のプロセスの定義には、図5に示すように、さらに小さな機能の定義や、場合分け等のプログラム記述がある。これらの定義やプログラム記述の中で、分岐やブロック化によってまとめられる実行の単位を、ここでは実行ブロックと呼ぶ。一つのプロセス中、処理の高速化を要求される実行ブロックがあれば、その反対に要求されない実行ブロックもある。シミュレータは、各実行ブロックについて、処理の高速化が要求されるかどうかを判断するためのプロファイリング情報を作成するためのものである。プロファイリング情報は、各実行ブロックに含まれる変数へのアクセス回数を含んでいる。すなわち、アクセス回数が一定以上であれば、その変数が属する実行ブロックについては処理の高速化が必要であると推定する。反対に、アクセス回数が一定以下であれば、その変数が属する実行ブロックについては処理の高速化は必要でないと推定する。このように、各ブロックに属する変数へのアクセス回数をプロファイリング情報として収集することにより、各ブロックに対し、処理の高速化が必要か否かを知ることが出来るようになる。そして、仮のソフトウェア実現候補内の実行ブロックに処理の高速化が必要なものがあると判断すれば、その実行ブロックをハードウェア実装への変更候補に設定する。また、仮のハードウェア実現候補内の実行ブロックに処理の高速化が必要でないものがあると判断すれば、その実行プロセスをソフトウェア実装への変更候補に設定する。

【0037】図1の協調合成システム2に含まれるシミュレータは、システム仕様記述からシミュレーションコードを作成してシミュレーションを実行出来るようにし、また、そのときに各変数がどの実行ブロックに属す

るかを示すスコープ情報を作成し、これを各実行ブロックの従属関係を示す木（ツリー）構造で管理する。木構造の各実行ブロックには変数のアクセス回数をカウントするカウンタが埋め込まれ、シミュレーションコードの実行に従って、このカウンタを加算していく。これにより、実行ブロック毎の変数のアクセス回数をカウントすることが出来るから、この結果をプロファイリング情報として出力する。図6にシミュレーションを行うときの状態を示す。

【0038】次に、上記のようにして得られたプロファイリング情報から、ソフトウェア実現候補の中のプロセスのうち所定の実行ブロックをハードウェア実装への変更候補に選び、また、ハードウェア実現候補の中のプロセスのうち所定の実行ブロックをソフトウェア実装への変更候補に選ぶ。以下、この選択手法について説明する。

【0039】（1）ソフトウェアからハードウェアへの入れ替え候補の選択

ソフトウェア実現候補のプロセスに含まれる実行ブロックのうち、シミュレーション実行後のアクセス回数の多い実行ブロックのランク付けを行い、高いランク（高アクセス回数）を持つ実行ブロックのそれぞれについて、ハードウェアとして切り出した場合のオーバーヘッドを加算したハードウェア増加面積を算出する。オーバーヘッドとは、ハードウェアとして切り出した場合に、余分に必要となるハードウェア部分（例えば、アドレス設定レジスタなど）をいう。この演算は、最高アクセス回数を持つ実行ブロックから最高アクセス回数の2分の1の回数を持つ実行ブロックまでを対象とする。

【0040】（2）ハードウェアからソフトウェアへの入れ替え候補の選択

ハードウェア実現候補のプロセスに含まれる実行ブロックのうち、シミュレーション実行後のアクセス回数の少ない実行ブロックのランク付けを行い、高いランク（低アクセス回数）を持つ実行ブロックのそれぞれについて、ソフトウェアとして切り出した場合のオーバーヘッドを減算したハードウェア減少面積を算出する。この演算は、最低アクセス回数を持つ実行ブロックから最高アクセス回数の2分の1の回数を持つ実行ブロックまでを対象とする。

【0041】上記（1）（2）で選択した入れ替え候補について、以下のアルゴリズムにより入れ替えを確定する。

【0042】（A）上記（2）の入れ替え候補の実行ブロック（以下、ソフトウェア実装変更候補モジュールと呼ぶ）を、ソフトウェア合成システム5（図1参照）を用いて処理し、ソフトウェア見積もりデータを取得する。ソフトウェア見積もりデータには、プログラムステップ数と実行時間が含まれる。

【0043】（B）上記（1）の入れ替え候補の実行ブ

ロック（以下、ハードウェア実装変更候補モジュールと呼ぶ）を、動作合成システム3（図1参照）を用いて処理し、ハードウェア見積もりデータを取得する。ハードウェア見積もりデータには、実装の面積と実行時間が含まれる。

【0044】（C）保持していた初期分割におけるソフトウェア見積もりデータ（初期見積もりデータ）とソフトウェア実装変更候補モジュールのソフトウェア見積もりデータとを比較する。比較により、面積改善率と速度改悪率を求める。前者の面積改善率は、ソフトウェア実装変更候補モジュールの実現により減少する面積の改善率であり、速度改悪率は、ソフトウェア実装変更候補モジュールの実現により遅くなる速度の改悪率である。その結果、面積改善率/速度改悪率が1.0以下のモジュールを非効率実装モジュールとし、これをソフトウェア実装変更候補モジュールから除去する。

【0045】（D）保持していた初期分割におけるハードウェア見積もりデータ（初期見積もりデータ）とハードウェア実装変更候補モジュールのハードウェア見積もりデータとを比較する。比較により、速度改善率と面積改悪率を求める。前者の速度改善率は、ハードウェア実装変更候補モジュールの実現により速くなる速度の改善率であり、面積改悪率は、ハードウェア実装変更候補モジュールの実現により増大する面積の改悪率である。その結果、速度改善率/面積改悪率が1.0以下のモジュールを非効率実装モジュールとし、これをハードウェア実装変更候補モジュールから除去する。

【0046】（E）ソフトウェア実装変更候補モジュールにおいて、面積改善率の大きなものから実装変更優先度を付与する。

【0047】（F）ハードウェア実装変更候補モジュールにおいて、速度改善率の大きなものから実装変更優先度を付与する。

【0048】（G）ハードウェア実装変更候補モジュールの実装変更優先度の高いものから順に参照し、その面積増加分を補うソフトウェア実装変更候補モジュールの組み合わせを探索する。つまり、面積が増えないように全体として処理速度が改善される組み合わせを探索する。この探索により、可能な限り実装変更優先度の高いものを組み合わせ、同一実装面積でさらに処理速度が改善される組み合わせを探る。

【0049】上記（G）の処理は、一定の面積内で速度が最小となる方向を目指す。この具体的な内容について図7～図9を参照して説明する。

【0050】図7は、（E）の処理を終えたときのソフトウェア実装変更候補モジュールのソート結果を示している。また、図8は、（F）の処理を終えたときのハードウェア実装変更候補モジュールのソート結果を示している。図7において、1列目の「H1」・・・はソフトウェア実装変更候補モジュールの識別番号、2列目の

「面積」と「実行時間」は、初期見積もりデータとして得られた当初のハードウェア部分での面積とその部分での実行時間、3列目の「ステップ数」と「実行時間」は、ソフトウェア実装変更候補モジュールについて

（A）の処理により得られたステップ数と実行時間、4列目の「減少面積」はソフトウェアからハードウェアへ変換した場合の減少面積、すなわち、2列目の面積から3列目のステップ数面積換算値を引いた値、5列目の「既変」は候補から実際に変換されたことを示すフラグ、6列目の「今回」は変更候補となったことを示すフラグである。また、図8において、1列目の「S1」・・・はハードウェア実装変更候補モジュールの識別番号、2列目の「ステップ数」と「実行時間」は、初期見積もりデータとして得られた当初のソフトウェア部分でのステップ数と実行時間、3列目の「面積」と「実行時間」は、ハードウェア実装変更候補モジュールについて（B）の処理により得られた面積と実行時間、4列目の「増加面積」はハードウェアへと変換した場合の増加面積すなわち、3列目の面積から2列目のステップ数面積換算値を引いた値、5列目の「既変」は候補から実際に変換されたことを示すフラグ、6列目の「今回」は変更候補となったことを示すフラグである。

【0051】図9は、図7及び図8のソート結果を使って上記（G）の処理を実際に行うフローチャートである。

【0052】変数（カウンタ）nは図8に示すハードウェア実装変更候補モジュールのポイントを示し、変数（カウンタ）mは図9に示すソフトウェア実装変更候補モジュールのポイントを示す。

【0053】ST1では、ハードウェア実装変更候補モジュールの中で最も速度改善値の高いものを指定する。ST2において、そのモジュールの増加面積を読む。ST3では、ソフトウェア実装変更候補モジュールの中で最も面積改善値の高いものを指定し、ST5でそのモジュールの減少面積を読み込む。

【0054】変数「比較ゲート数」は、ソフトウェア実装変更候補モジュールの減少面積の足し込んだ値を示す。ST5において読み込んだソフトウェア実装変更候補モジュールの減少面積がST7において比較ゲート数に加算されていく。その加算値がST2で読み込んだハードウェア実装変更候補モジュールの増加面積に等しくなるまで、ST9を経由してソフトウェア実装変更候補モジュールの減少面積が比較ゲート数に加算されていく。ST8においてこれらの値が等しくなると、ST10において、該当のソフトウェア実装変更候補モジュールの「既変」フラグをオンにし、実際のソフトウェア実装分として確定する。また、ST11において、nで指定しているハードウェア実装変更候補モジュールの「既変」フラグをオンにして、実際のハードウェア実装分として確定する。なお、実装面積の均衡をとる場合に、ハ

ードウェア実装変更候補モジュールの実装採用による速度改善は、速度改善率の高いものから実装採用となるため、その組み合わせとなるソフトウェア実装変更候補モジュールの実装採用による速度改悪を十分にカバーする。

【0055】以上の $n_2 \sim n_{11}$ までの動作を n が所定値になるまで連続して処理することにより、チップ面積を増加させずに可能な限り処理速度の向上を図ることが出来る。

【0056】なお、ST2において1つのハードウェア実装変更候補モジュールの増加面積を読み出すようにしているが、このステップで上位から複数のハードウェア実装変更候補モジュールの面積加算値を読み出し、これに対応する面積減少値を持つソフトウェア実装変更候補モジュールを選ぶようにすることも出来る。また、反対に、最初にソフトウェア実装変更候補モジュールの面積減少値を読み出し、これに対応する面積増加値を持つハードウェア実装変更候補モジュールを選ぶようにすることも出来る。

【0057】また、上記の処理を一通り行った後、再度、最初から同じ動作を繰り返すことも可能である。何度も繰り返すことにより、さらに、速度向上を図ることが出来るようになる。さらに、チップ面積を増やすことが可能な場合には、その面積に対応する増加面積を有するハードウェア実装変更候補モジュールをハードウェア実装分にすることが出来る。

【0058】以上の処理により、仮のソフトウェア実装変更候補と仮のハードウェア実現候補とに初期分割したプロセスにおいて、チップ面積を大きくすることなく、速度改善出来るように自動的に各プロセス内の実行ブロックに対して、ソフトウェア実装への変更またはソフトウェア実装への変更を行うことが出来る。

【0059】図10～図12は、本発明の他の実施形態の動作を示している。図7～図9は、一定の面積内で実行速度が最小となる方向に各モジュールを変更する場合の操作内容を示しているが、一定の速度内で面積が最小となる方向に各モジュールを変更することも可能である。図10～図12はこの場合の動作を示す。

【0060】すなわち、図10に示すように、面積改善値でソートした各ソフトウェア実装変更候補モジュールに、ハードウェアからソフトウェアへ変換するときの増加時間、すなわち、2列目の実行時間から3列目の実行時間を引いた値を付加し、図11に示すように、速度改善値でソートした各ハードウェア実装変更候補モジュールに、ソフトウェアからハードウェアへ変換するときの減少時間、すなわち、3列目の実行時間から2列目の実行時間を引いた値を付加した状態で、図12のフローチャートを実行する。

【0061】図12では、変数(カウンタ) n で図10のソフトウェア実装変更候補モジュールを先頭から順に

ポインティングし、変数(カウンタ) m で図11のハードウェア実装変更候補モジュールを先頭から順にポインティングする。ST21で n で示されるソフトウェア実装変更候補モジュールの増加時間に均衡がとれる減少時間を持つハードウェア実装変更候補モジュールを図11から探し、それらをそれぞれソフトウェア実装分とハードウェア実装分とする。あとの処理は図9と同様である。

【0062】このように、上の2つの実施形態では、それぞれ、実装面積や実行時間の均衡をとりながらモジュールを交換して、一定の面積で最も速度が最小となる分割点や一定の速度で最も面積が最小となる分割点を自動的に求めることが出来る。

【0063】

【発明の効果】本発明によれば、最初に、システム仕様記述である実行単位のプロセスの全部を仮のハードウェア実現候補と仮のソフトウェア実現候補に自動分割しているため、分割に人間の経験や勘を必要せず、熟練技術者でなくとも分割が可能である。また、仮の分割点から、それぞれへの変更候補を選択して、チップ面積と実行速度の改善率を評価し、上記分割点の修正を行うようにしているため、システムの最適分割点を自動的に短時間で求めることが出来る。このため、システムの同時並行的開発が可能となる。

【0064】また、上記仮の候補への自動分割を優先度情報の有無により行うことにより、大きく誤った仮分割となることがなく、分割そのものも非常に簡単となり短時間で出来る。

【0065】また、1つのプロセスには複数の実行ブロックがあり、各ブロックによりソフトウェアが適切なものやハードウェアが適切なものがあるため、変更候補の最小単位をプロセス全体ではなくプロセス内の実行ブロックとすることで、より適切な分割が可能になる。

【0066】また、変数へのアクセスが多いときはハードウェア化がより適切なものと考えられ、反対に変数へのアクセスが少ないときはソフトウェア化が適切と考えられるから、プロファイリング情報として、各プロセスに含まれる実行ブロック内の変数へのアクセス回数を含ませれば、変更候補の選択がより正しい方向のものとなる。また、速度改善値の高いハードウェア実装変更候補とそれに実装面積の均衡をとり得るソフトウェア実装変更候補の組を選び、これらをそれぞれハードウェア実装分とソフトウェア実装分に変換する分割点修正を行うことで、一定の面積で速度が最小となる分割点を自動的に見いだすことが出来る。

【0067】また、面積改善値の高いソフトウェア実装変更候補とそれに実行時間の均衡をとり得るハードウェア実装変更候補の組を選び、これらをそれぞれソフトウェア実装分とハードウェア実装分に変換する分割点修正を行うことで、一定の時間で面積が最小となる分割点を

10

20

30

40

50

自動的に見いだすことが出来る。

【図面の簡単な説明】

【図 1】本発明の実施形態である、ハードウェアとソフトウェアの混在するシステムの設計支援装置の構成図。

【図 2】プロセスの集まりで書かれるシステム仕様記述の一例を示す図。

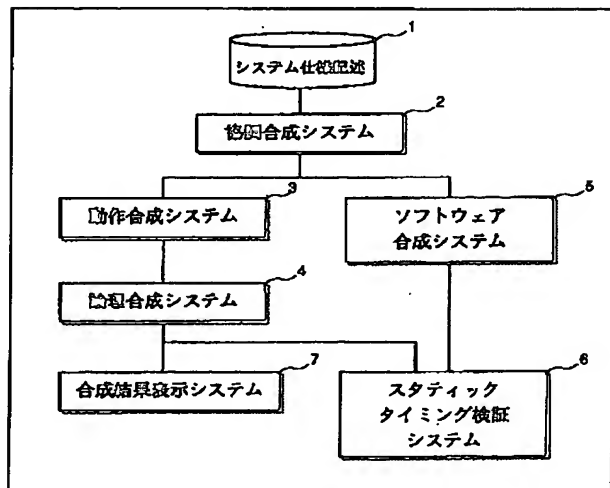
【図 3】システム仕様記述におけるプロセスの優先度指定を示す図。

【図 4】プロセスの初期分割例を示す図。

【図 5】プロセス内の実行ブロックを示す図。

【図 6】シミュレータによるシミュレーション動作を説明する図。

【図 1】



システム構成

【図 2】

```

process    sample
input  int  iData;           : プロセス名
output int  oData=0;         : 入力データ端子
ctrl_input Str1;             : 出力データ端子
ctrl_output ockStr=FALSE;    : 制御入力端子
{
  int counter=0;
  Mrol main(Str1){           : 通常の初期スタート (Str1がTRUE
    ockStr = TRUE;           : になると実行される)
    if ((++counter) > iData) : 条件判別文
      counter = 0;          : 代入文
      oData = iData + counter; : 代入文
  }
}
  
```

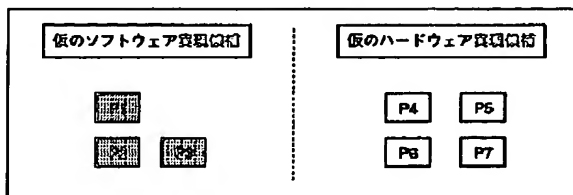
【図 3】

```

....
process PROC[A][0] abc; ..... 優先度: 0
....
....
process PROC[B][1] xyz; ..... 優先度: 1
....
  
```

システム仕様記述におけるプロセスの優先度指定

【図 4】



プロセスの初期分割

* 【図 7】ソフトウェア実装変更候補モジュールのソート結果を示す図。

【図 8】ハードウェア実装変更候補モジュールのソート結果を示す図。

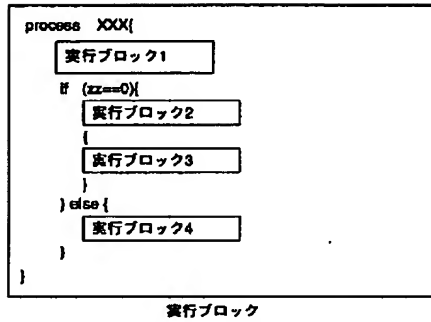
【図 9】速度が最小となる方向でモジュール変換を行う手順を示すフローチャート。

【図 10】ソフトウェア実装変更候補モジュールのソート結果を示す図。

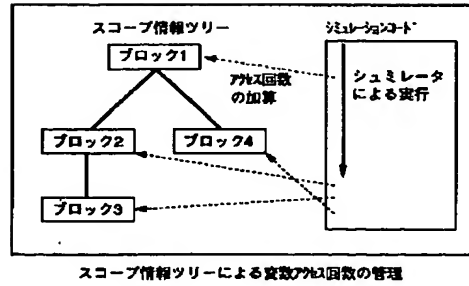
【図 11】ハードウェア実装変更候補モジュールのソート結果を示す図。

* 【図 12】面積が最小となる方向でモジュール変換を行う手順を示すフローチャート。

【図5】



【図6】



【図8】

【図7】

ソフトウェア実装変更候補モジュール

H1	面積 実行時間	ステップ数 実行時間	減少面積	既 変	今 回
H2	面積 実行時間	ステップ数 実行時間	減少面積	既 変	今 回
H3	面積 実行時間	ステップ数 実行時間	減少面積	既 変	今 回
HM	面積 実行時間	ステップ数 実行時間	減少面積	既 変	今 回
HM+1	面積 実行時間	ステップ数 実行時間	減少面積	既 変	今 回
HM+2	面積 実行時間	ステップ数 実行時間	減少面積	既 変	今 回

大
面積改善値

当初計算されたデータ
(初期見積りデータ)

変更した場合の
換算データ

変更された事を
示すフラグ

今回変更候補とな
った事を示すフラグ

ハードウェア実装変更候補モジュール

S1	ステップ数 実行時間	面積 実行時間	増加面積	既 変	今 回
S2	ステップ数 実行時間	面積 実行時間	増加面積	既 変	今 回
S3	ステップ数 実行時間	面積 実行時間	増加面積	既 変	今 回
SN	ステップ数 実行時間	面積 実行時間	増加面積	既 変	今 回
SN+1	ステップ数 実行時間	面積 実行時間	増加面積	既 変	今 回
SN+2	ステップ数 実行時間	面積 実行時間	増加面積	既 変	今 回

大
速度改善値

当初計算されたデータ
(初期見積りデータ)

変更した場合の
換算データ

変更した場合の
増加面積

【図10】

【図11】

ソフトウェア実装変更候補モジュール

H1	面積 実行時間	ステップ数 実行時間	増加時間	既 変	今 回
H2	面積 実行時間	ステップ数 実行時間	増加時間	既 変	今 回
H3	面積 実行時間	ステップ数 実行時間	増加時間	既 変	今 回

大
面積改善値

HM	面積 実行時間	ステップ数 実行時間	増加時間	既 変	今 回
HM+1	面積 実行時間	ステップ数 実行時間	増加時間	既 変	今 回
HM+2	面積 実行時間	ステップ数 実行時間	増加時間	既 変	今 回

当初計算された
データ

変更した場合の
換算データ

変更された事を
示すフラグ

今回変更候補とな
った事を示すフラグ

ハードウェア実装変更候補モジュール

S1	ステップ数 実行時間	面積 実行時間	減少時間	既 変	今 回
S2	ステップ数 実行時間	面積 実行時間	減少面積	既 変	今 回
S3	ステップ数 実行時間	面積 実行時間	減少面積	既 変	今 回
SN	ステップ数 実行時間	面積 実行時間	減少面積	既 変	今 回
SN+1	ステップ数 実行時間	面積 実行時間	減少面積	既 変	今 回
SN+2	ステップ数 実行時間	面積 実行時間	減少面積	既 変	今 回

大
速度改善値

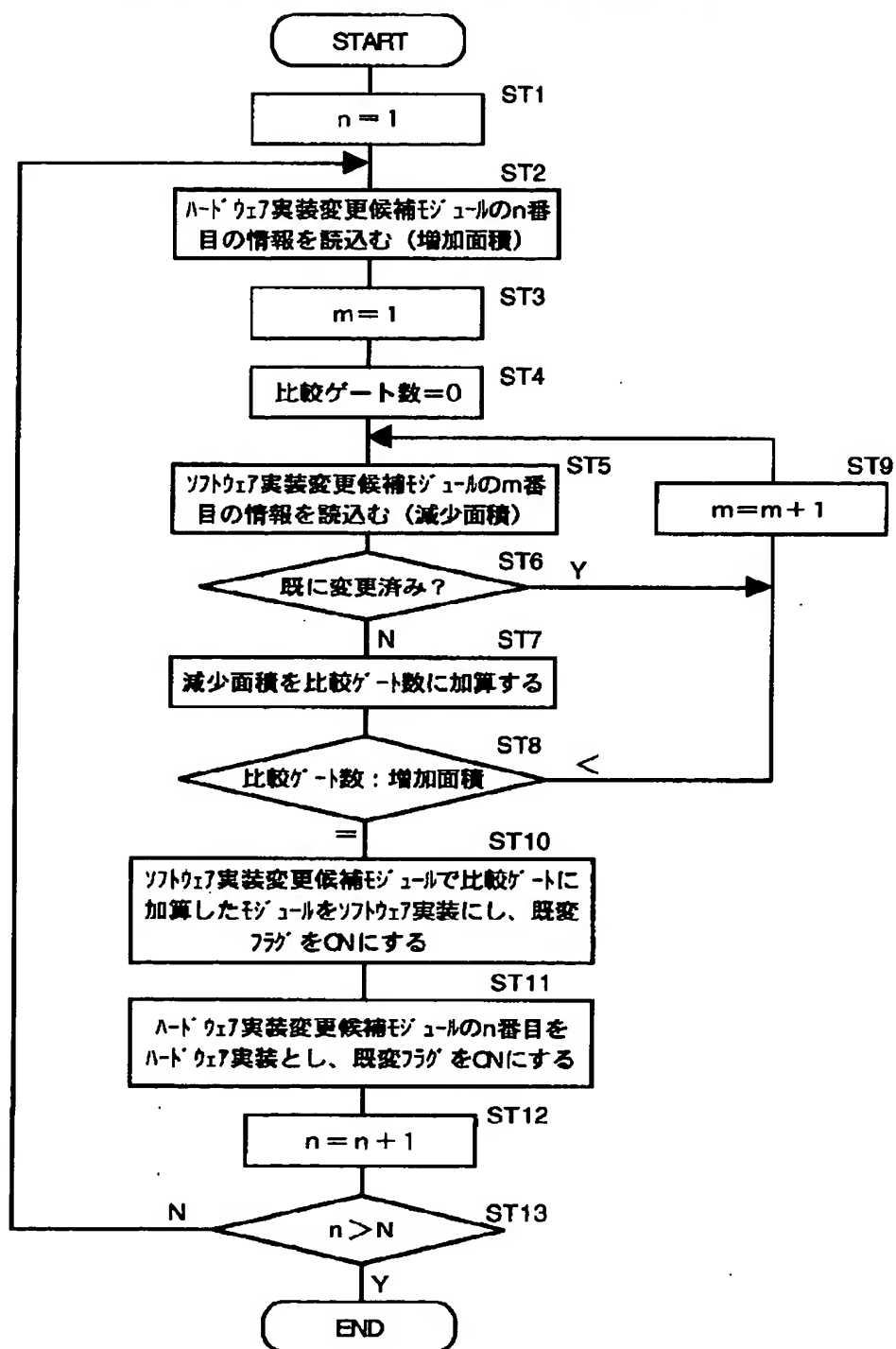
当初計算された
データ

変更した場合の
換算データ

変更した場合の
増加面積

【図 9】

一定の面積内で、速度が最小となる方向で変換する手順



【図12】

面積制約がない場合

一定の速度内で、面積が最小となる方向で変換する

